# Introducción a Control de Versiones de Código

# ¿Qué es el "control de versiones" y por qué es importante?

El control de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a lo largo del tiempo para que pueda recuperar versiones específicas más adelante.

Un sistema de control de versiones (VCS) es algo muy inteligente de usar.

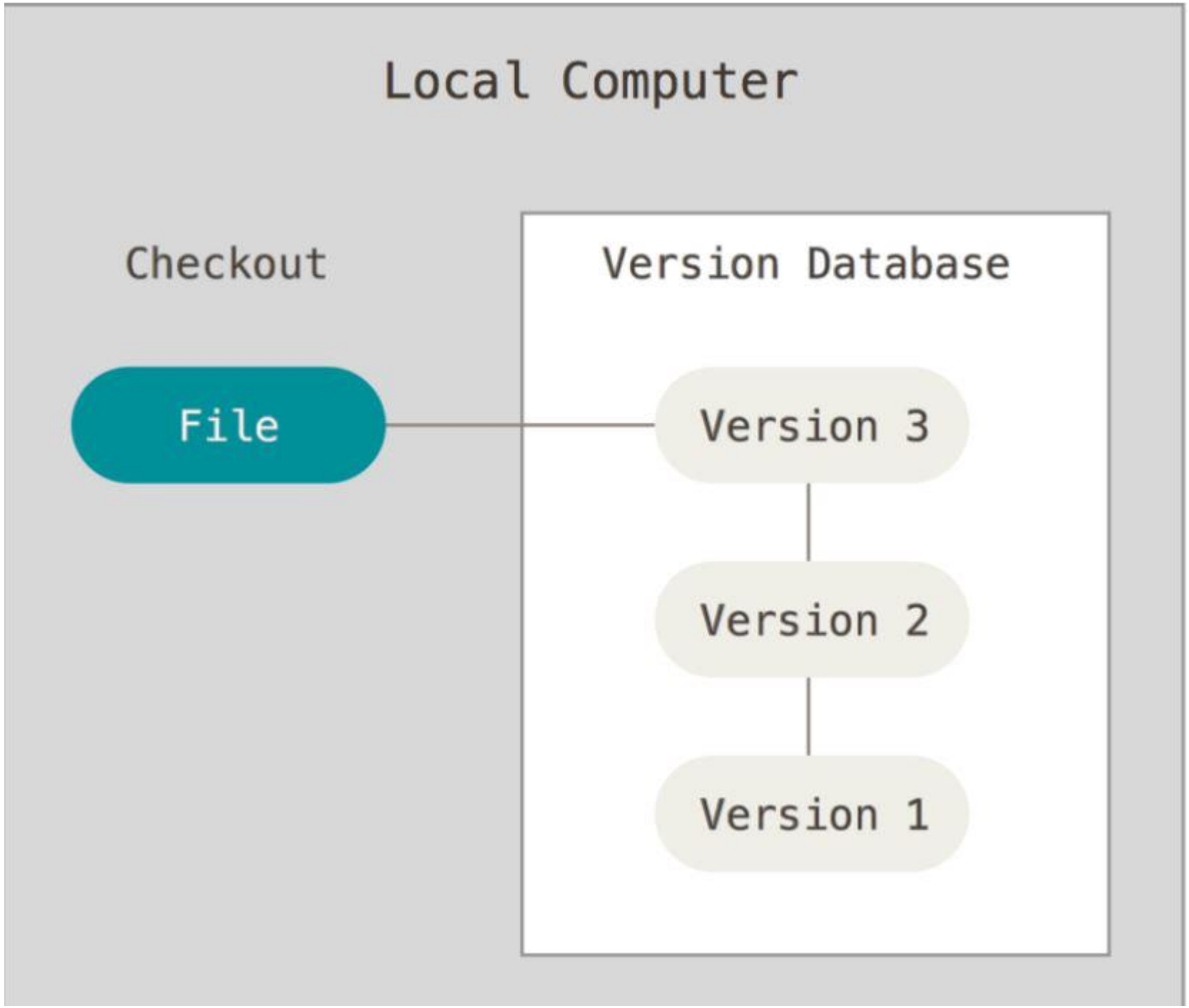
- Le permite revertir los archivos seleccionados a un estado anterior,
- Revertir todo el proyecto al estado anterior
- Comparar los cambios a lo largo del tiempo
- Ver quién modificó por última vez algo que podría estar causando un problema
- Quién presentó un problema y cuándo
- Y mucho más.

Usar un VCS también generalmente significa que si arruinas las cosas o pierdes archivos, puedes recuperarte fácilmente.

El método de elección de control de versiones de muchas personas es copiar archivos en otro directorio (quizás un directorio con sello de tiempo, si son inteligentes).

Este enfoque es muy común porque es muy simple, pero también es increíblemente propenso a errores. Es fácil olvidar en qué directorio se encuentra y escribir accidentalmente en el archivo incorrecto o copiar sobre archivos que no quiere.

Para lidiar con este problema, los programadores desarrollaron hace mucho tiempo VCS locales que tenían una base de datos simple que mantenía todos los cambios en archivos bajo control de revisión.



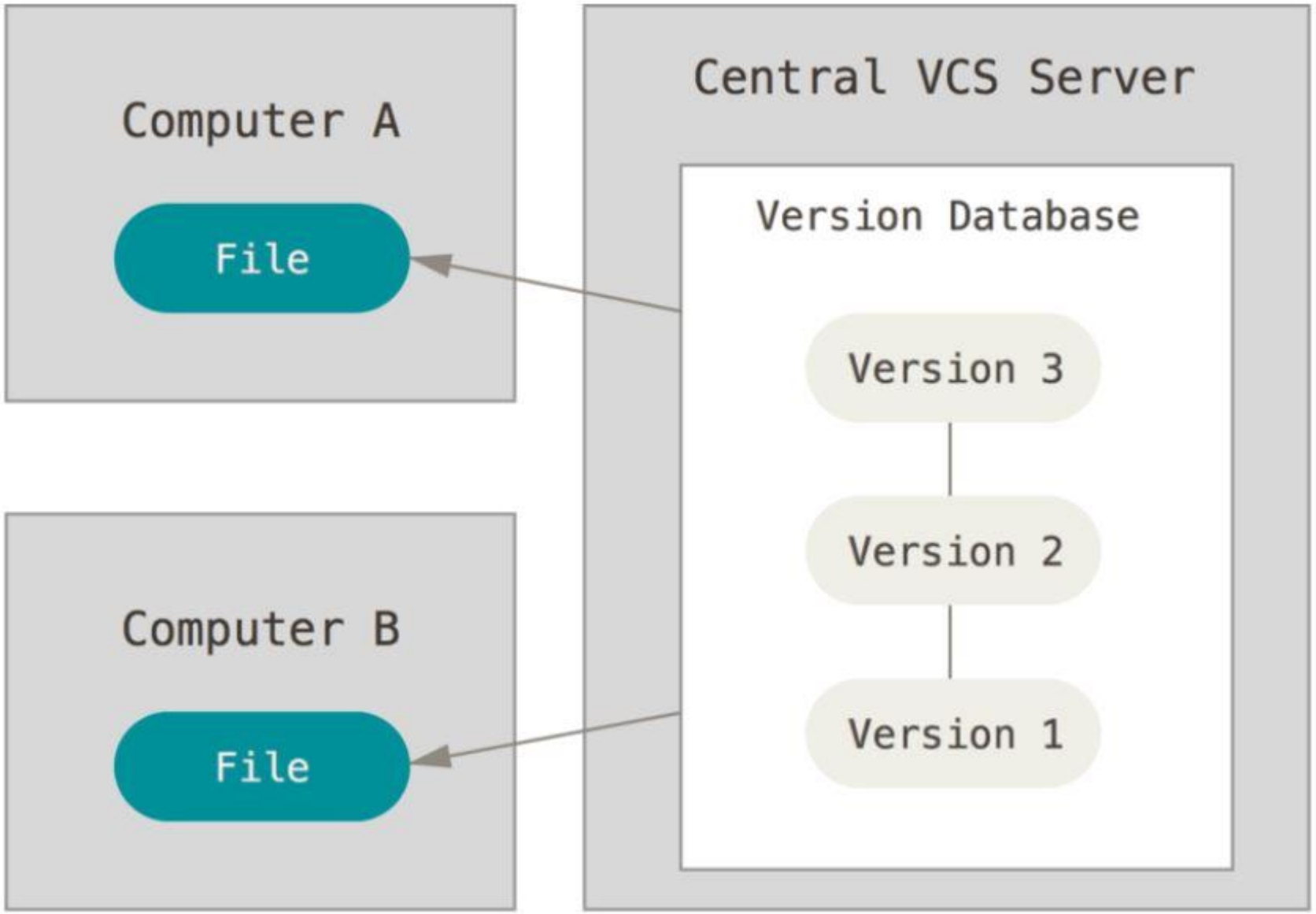

Una de las herramientas VCS más populares fue un sistema llamado RCS, que todavía se distribuye con muchas computadoras hoy en día. RCS funciona manteniendo conjuntos de parches (es decir, las diferencias entre archivos) en un formato especial en el disco; luego puede volver a crear el aspecto de cualquier archivo en cualquier momento agregando todos los parches.

# Sistemas centralizados de control de versiones

El siguiente problema importante que enfrentan las personas es que necesitan colaborar con los desarrolladores en otros sistemas.

Para solucionar este problema, se desarrollaron sistemas de control de versiones centralizados (CVCS). Estos sistemas (como CVS, Subversion y Perforce) tienen un único servidor que contiene todos los archivos versionados y una cantidad de clientes que extraen archivos de ese lugar central.

Durante muchos años, este ha sido el estándar para el control de versiones.

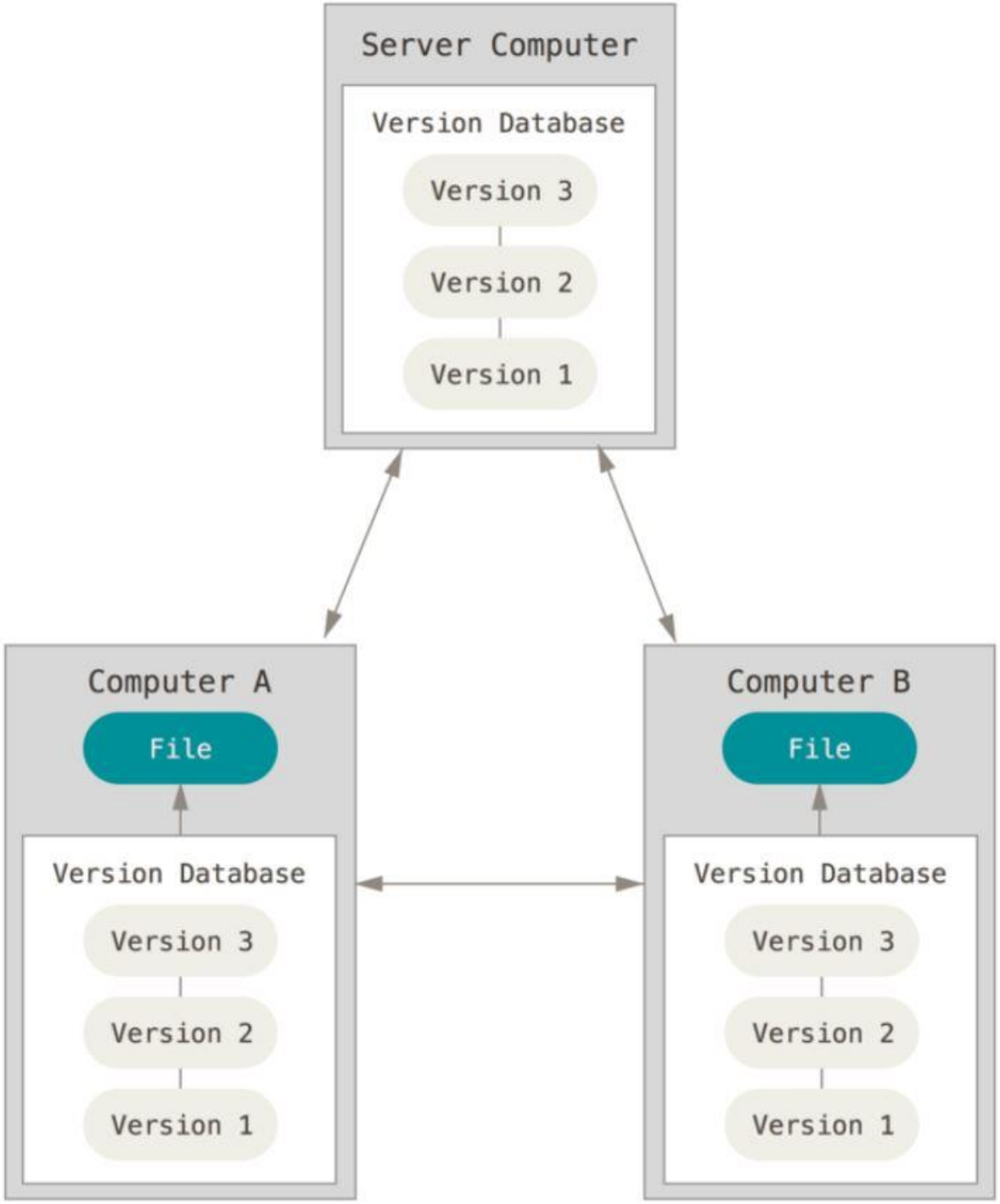



Esta configuración ofrece muchas ventajas, especialmente sobre los VCS locales. Por ejemplo, todos saben en cierto grado lo que todos los demás en el proyecto están haciendo. Sin embargo, esta configuración también tiene algunas desventajas serias. Lo más obvio es el único punto de falla que representa el servidor centralizado…….

Los sistemas VCS locales sufren este mismo problema: cada vez que tiene todo el historial del proyecto en un solo lugar, corre el riesgo de perderlo todo.

# Sistemas distribuidos de control de versiones

Aquí es donde intervienen los sistemas distribuidos de control de versiones (DVCS). En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no solo revisan la última instantánea de los archivos; más bien, reflejan completamente el repositorio, incluida su historia completa. Por lo tanto, si algún servidor muere, y estos sistemas estaban colaborando a través de ese servidor, cualquiera de los repositorios de clientes puede copiarse nuevamente en el servidor para restaurarlo. Cada clon es realmente una copia de seguridad completa de todos los datos.




Además, muchos de estos sistemas tratan bastante bien con tener varios repositorios remotos con los que pueden trabajar, por lo que puede colaborar con diferentes grupos de personas de diferentes maneras simultáneamente en el mismo proyecto. Esto le permite configurar varios tipos de flujos de trabajo que no son posibles en sistemas centralizados, como los modelos jerárquicos.

# Como trabaja ??

Conceptualmente, la mayoría de los otros sistemas almacenan información como una lista de cambios basados en archivos. Estos otros sistemas (CVS, Subversion, Perforce, Bazaar, etc.) piensan en la información que almacenan como un conjunto de archivos y los cambios realizados en cada archivo a lo largo del tiempo (esto se describe comúnmente como control de versión basado en delta

Git no piensa ni almacena sus datos de esta manera. En cambio, Git piensa en sus datos más como una serie de instantáneas de un sistema de archivos en miniatura. Con Git, cada vez que confirma o guarda el estado de su proyecto, Git básicamente toma una fotografía de cómo se ven todos sus archivos en ese momento y almacena la referencia a esa instantánea. Para ser eficiente, si los archivos no han cambiado, Git no almacena el archivo nuevamente, solo un enlace al archivo idéntico anterior que ya ha almacenado. Git piensa en sus datos más como una secuencia de instantáneas

# Ejercicios:

1- Defina un usuario en Github
2- Descargue e instale Github Desktop ( https://desktop.github.com/)
3- Configure su cuenta de GitHub con su Desktop
4- Crear un repositorio en GitHub remoto (GitHub.com)
5 – Crear otro repositorio de prueba a través del GitHub Desktop
6 – Subir al remoto un archivo (Si contien código MEJOR) 

Referencia:
Para definir un usuario en GitHub:
https://github.com/
Guia en castellano para configurar el GitHub Desktop :
https://help.github.com/es/desktop/getting-started-with-github-desktop/installing-and-authenticating-to-github-desktop